

本节内容

串 存储结构

知识总览

串的存储结构

顺序存储

链式存储

基于顺序存储实现基本操作

串的顺序存储

结合顺序表的知识思考优缺点

```
#define MAXLEN 255
```

```
typedef struct{
```

```
    char ch[MAXLEN];
```

```
    int length;
```

```
}SString;
```

静态数组实现
(定长顺序存储)

//预定义最大串长为255

//每个分量存储一个字符

//串的实际长度

```
typedef struct{
```

```
    char *ch;
```

```
    int length;
```

```
}HString;
```

动态数组实现
(堆分配存储)

//按串长分配存储区, ch指向串的基地址

//串的长度

```
HString S;
```

```
S.ch = (char *) malloc(MAXLEN * sizeof(char));
```

```
S.length = 0;
```

用完需要手动free

S.ch →

内存

ch[0]

ch[1]

ch[2]

ch[3]

ch[4]

...

...

ch[254]

length

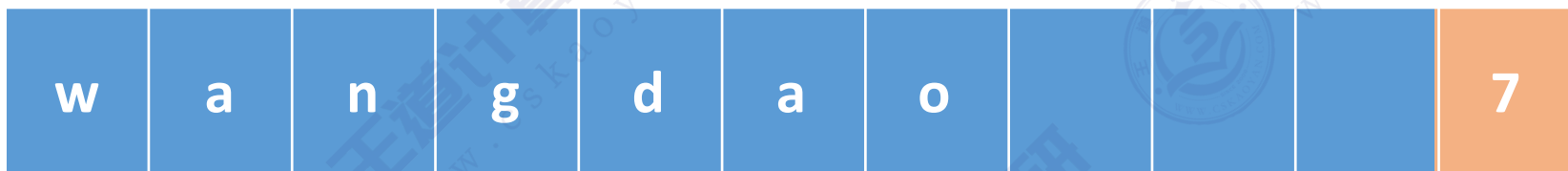
分配连续的存储空间,
每个 char 字符占 1B

串的实际长度

串的顺序存储

char ch[10]

方案一:



变量Length

ch[0]充当 Length

方案二:



优点: 字符的位序
和数组下标相同

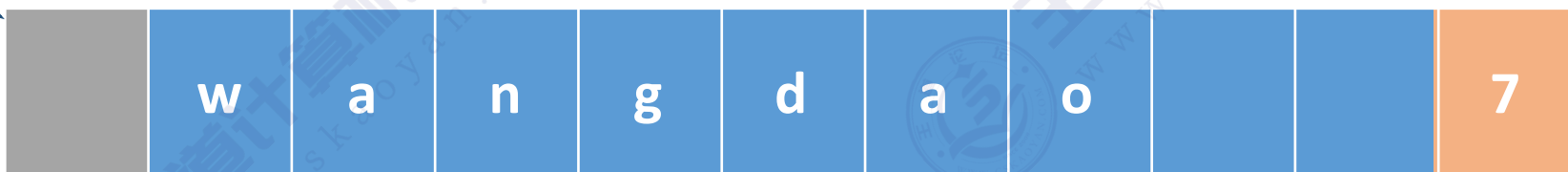
方案三:



没有Length变量,
以字符'\0'表示结尾
(对应ASCII码的 0)

ch[0]废
弃不用

方案四:
(教材)



变量Length

串的链式存储

结合链表的知识
思考优缺点

```
typedef struct StringNode{  
    char ch;    //每个结点存1个字符  
    struct StringNode * next;  
}StringNode, * String;
```



存储密度低：每个字符1B，每个指针4B

```
typedef struct StringNode{  
    char ch[4];    //每个结点存多个字符  
    struct StringNode * next;  
}StringNode, * String;
```



存储密度提高

基本操作的实现

ch[0]废弃不用

方案四：
(教材)



变量Length

```
#define MAXLEN 255 //预定义最大串长为255
typedef struct{
    char ch[MAXLEN]; //每个分量存储一个字符
    int length; //串的实际长度
}SString;
```

StrAssign(&T,chars): 赋值操作。把串T赋值为chars。

StrCopy(&T,S): 复制操作。由串S复制得到串T。

StrEmpty(S): 判空操作。若S为空串，则返回TRUE，否则返回FALSE。

StrLength(S): 求串长。返回串S的元素个数。

ClearString(&S): 清空操作。将S清为空串。

DestroyString(&S): 销毁串。将串S销毁（回收存储空间）。

Concat(&T,S1,S2): 串联接。用T返回由S1和S2联接而成的新串

基本操作的实现

SubString(&Sub,S,pos,len): 求子串。用Sub返回串S的第pos个字符起长度为len的子串。

S.ch="wangdao"
S.length=7

S		w	a	n	g	d	a	o		
	0	1	2	3	4	5	6	7	8	9

//求子串

```
bool SubString(SString &Sub, SString S, int pos, int len){
```

```
    //子串范围越界
```

```
    if (pos+len-1 > S.length)
```

```
        return false;
```

```
    for (int i=pos; i<pos+len; i++)
```

```
        Sub.ch[i-pos+1] = S.ch[i];
```

```
    Sub.length = len;
```

```
    return true;
```

```
}
```

```
#define MAXLEN 255
```

```
typedef struct{
```

```
    char ch[MAXLEN];
```

```
    int length;
```

```
}SString;
```

//预定义最大串长为255

//每个分量存储一个字符

//串的实际长度

基本操作的实现

StrCompare(S,T): 比较操作。若 $S>T$, 则返回值 >0 ; 若 $S=T$, 则返回值 $=0$; 若 $S<T$, 则返回值 <0 。

S.ch="wangdao"
S.length=7

S.ch →

	w	a	n	g	d	a	o		
0	1	2	3	4	5	6	7	8	9

//比较操作。若 $S>T$, 则返回值 >0 ; 若 $S=T$, 则返回值 $=0$; 若 $S<T$, 则返回值 <0

```
int StrCompare(SString S, SString T) {  
    for (int i=1; i<=S.length && i<=T.length; i++){  
        if (S.ch[i]!=T.ch[i])  
            return S.ch[i]-T.ch[i];  
    }  
    //扫描过的所有字符都相同, 则长度长的串更大  
    return S.length-T.length;  
}
```

T1

	w	b	n	g	d	a	o		
0	1	2	3	4	5	6	7	8	9

T2

	w	a	n	g					
0	1	2	3	4	5	6	7	8	9

基本操作的实现

Index(S,T): 定位操作。若主串S中存在与串T值相同的子串，则返回它在主串S中第一次出现的位置；否则函数值为0。

S.ch="wangdao"
S.length=7

S.ch →

	w	a	n	g	d	a	o		
0	1	2	3	4	5	6	7	8	9

```
int Index(SString S, SString T){  
    int i=1, n=StrLength(S), m=StrLength(T);  
    SString sub;    //用于暂存子串  
    while(i<=n-m+1){  
        SubString(sub, S, i, m);  
        if(StrCompare(sub, T)!=0) ++i;  
        else return i; //返回子串在主串中的位置  
    }  
    return 0; //S中不存在与T相等的子串  
}
```

T

	g	d	a						
0	1	2	3	4	5	6	7	8	9

知识回顾与重要考点

串的存储结构

顺序存储

静态数组

```
#define MAXLEN 255
```

//预定义最大串长为255

```
typedef struct{
```

```
    char ch[MAXLEN];
```

//每个分量存储一个字符

```
    int length;
```

//串的实际长度

```
}SString;
```

动态数组

malloc、free

链式存储

可让每个结点存多个字符，没有字符的位置用'#'或'\0'补足

王道教材采用
——静态数组

ch[0] 废弃不用

方案四：
(教材)

w a n g d a o

7

变量Length

基本操作的实现

求子串: bool SubString(SString &Sub, SString S, int pos, int len)

串的比较: int StrCompare(SString S, SString T)

求串在主串中的位置: int Index(SString S, SString T)